Making Corpus Querying Ready for the Future: Challenges and Concepts

Markus Gärtner and Jonas Kuhn
Institute for Natural Language Processing
University of Stuttgart
Germany
{markus.gaertner,jonas.kuhn}@ims.uni-stuttgart.de

Abstract

This paper presents work in progress on a project to improve usability of corpus query systems, and in particular their query languages, for user groups that are easily intimidated by overly complex tools. Having evaluated a series of existing corpus query systems, we expand the existing catalog of (informal) requirements for such query systems with new findings. We hint at several conceptual shortcomings of modern corpus query languages and proceed by introducing the first design draft of our own hybrid approach. While doing so we also present two assumptions for future trends in the design and user interaction of corpus query systems.

1 Introduction

Ever since corpora[1] started to play a role in natural language processing or other disciplines, the need for corpus query systems (CQS) and their assorted corpus query languages (CQL[2]) also arose. These kind of systems are there to play a very straightforward, yet extremely important role. That is, they are meant to answer questions (queries) concerning a corpus that a researcher is interested in. Such queries can come in vastly different levels of complexity. Consider for example the following (made up) kinds of information a researcher might want from a corpus:

(1) Sentences with the word "elephant".

(2) Sentences that contain the word "elephant" regardless of inflection or capitalization.

(3) Sentences where "elephant" is the logical subject and not preceded by an adjective.

(4) Instances of a mention of a person, which is discourse-new, appearing adjacent to a preceding tonally prominent adjective and having not been marked tonally prominent itself. Include only persons that are not mentioned again in the same sentence but at least twice but no more than five times in the remainder of the same chapter.

Example (1) is a trivial string matching task, while (2) and (3) can be answered by directly accessing lemmatized, tagged or parsed content. Those are the kinds of questions typical corpus query systems are able to answer these days. Some systems add a bit of expressive power to combine different annotations layers or even provide rudimentary support for querying information from different modalities. But usually their coverage ends well before questions such as (4) can be properly answered.

The last example (4) is however slightly exaggerated in terms of complexity. This is done intentionally to emphasize the importance of CQS implementations as one of the enablers of corpus-based research, with a focus on future trends and upcoming (potential) questions. Especially on the points of contact between modalities there is still a lot of room for questions that for example take into account the rich multitude of phonetic annotations in combination with classic textual data.

The context of this paper is a project where we are faced with providing query access to a diverse set of richly annotated corpus resources of varying modality and vastly different annotation layers. Our user group comprises to a large extent researchers with only a minor technical background, making ease of use a top priority. In the remainder of this paper we discuss potential solutions and refine existing requirements for successful CQL designs. Finally we also introduce the first draft of

[1] For reasons of simplicity we apply the definition of corpora being collections of utterances without any special requirements regarding their modality or physical form.

[2] Not to be confused with the commonly used abbreviations for Cassandra Query Language or Contextual Query Language

Proceedings of the 14th Conference on Natural Language Processing (KONVENS 2018)
Vienna, Austria – September 19-21, 2018

our own hybrid CQL approach that aims at combining favorable concepts from different (corpus) query systems.

2 Motivation for Dedicated CQS

Today a plethora of industry grade solutions for data management and subsequent querying of its content exist. They include technologies such as relational databases with the query flagship *Structured Query Language (SQL)*[3], graph databases such as *Neo4j*[4], systems for indexing, e.g. *Apache Lucene*[5] and the *OWL2 Web Ontology Language*[6] for describing content in the Semantic Web and its associated query language *SPARQL*[7].

Having this catalog of readily available solutions begs the question of why they are not adopted as the main technologies for composing, managing and querying corpora. Jarke and Vassiliou (1985) already emphasized the division of query language evaluation criteria into two main categories: those of usability and functionality. While functionality is a purely technical aspect where features such as expressiveness (cf. (Lai and Bird, 2010) for a collection of requirements for treebank query tools), efficiency and others can be universally verified (e.g. in the work by Frick et al. (2012) for comparing several CQL implementations), the matter is much more complicated for usability. The question of usability is highly subjective and relies heavily on the target group of users for which it is to be answered.

The question of "Who would use a CQS?" can be answered very pragmatically with the statement "Everybody with interest in a corpus' content", which typically includes people from the fields of (computational) linguistics as well as (digital) humanities. This means that there is a certain disparity to be expected in the technical background knowledge and skills of this user group and the users of above systems, which usually are trained software engineers. This is in line with the usability considerations stated in Mueller (2010) when taking the perspective of a scholarly environment.

While integrations with those technologies have been done (for instance Burchardt et al. (2008) for OWL), there exist no widespread adaptations of them for use as a CQL front-end. From this the still existing need for CQS implementations tailored to the above mentioned user groups concludes.

3 Related Work

Two of the existing implementations we are aware of come closest to fulfilling above needs[8]:

ANNIS (Krause and Zeldes, 2014) is a web-based CQS that combines a relational database (SQL) as storage with the graph-based modeling toolkit SALT (Zipser, 2009). It provides a very flexible visual front-end that features several visualizations optimized to display certain linguistic phenomena. Its query language AQL uses a syntax of individual terms that express constraints for distinct elements or relations between the elements defined in other terms. Terms are then joined via the logical operators of either conjunction or disjunction. This allows it a high degree of expressiveness, but also makes complex (structural) queries very difficult to read. Relying on a relational database also poses certain disadvantages, since especially structural queries with closures are traditionally time consuming for those databases, hampering scalability severely. The recently developed graphANNIS (Krause et al., 2016) however provides an alternative to the previous database solution by serving as a graph database and eliminating some of the issues.

ICARUS[9] (Gärtner et al., 2013; Gärtner et al., 2015) on the other hand is a CQS implementation for local desktop use. Its storage is memory-based and as such introduces a bottleneck when operating with very large corpora. The number of different visualization modes is lower compared to ANNIS, but customizability of individual visualizations by the user is much higher. Unlike ANNIS it uses a hybrid syntax with brackets to express structural properties (nesting indicates dominance) and comma-separated term lists for local constraints inside individual nodes.

The following examples (5) and (6) illustrate the comparison between a simple structural query expressed in ANNIS and ICARUS syntax. Both queries match a node whose part-of-speech tag is

[3] Standardized under ISO/IEC 9075:2016 "Information technology – Database languages – SQL"

[4] `https://neo4j.com/`

[5] `https://lucene.apache.org/`

[6] `https://www.w3.org/TR/owl2-overview/`

[7] `https://www.w3.org/TR/sparql11-overview/`

[8] We did not include KoralQuery (Bingel and Diewald, 2015) as an implementation of CQLF (Banski et al., 2016) in this section, as it is meant as interchange protocol between CQL dialects and not for manual query construction by users.

[9] Interactive platform for Corpus Analysis and Research tools, University of Stuttgart

```
#vp:[cat="VP"] & #v: [pos="V"] & #np:[cat="NP"] & #pp:[cat="PP"]
& #vp >* #v & #vp >* #np & #vp >* #pp & #v >@r #vr & #np >@l #npl
& #vr .1 #npl & pp >@l ppl & npl .1 ppl
```

Figure 1: TigerSearch (Lezius, 2002) query to find verb phrases that contain a verb immediately followed by a noun phrase that is immediately followed by a prepositional phrase. Example by Lai and Bird (2010)

```
FIND sentence WITH vp,np,pp AS phrase AND v AS token WHERE [vp:
cat=VP [v: pos=V][np: cat=NP][pp: cat=PP]] AND adjacent(v,np,pp)
```

Figure 2: Mock-up of a possible hybrid-style query equivalent in content to the one in Figure 1.

categorized as `NP` that dominates a `PP` node:

(5) cat=”NP” & cat=”PP” & #1 > #2

(6) [pos=”NP” [pos=”PP”]]

Both systems allow the construction of queries textually as well as graphically (see Figure 3) with elaborate editors. Their overall designs however are not amendable to a degree that would make them usable for answering queries such as the one in Example (4) on huge corpora.

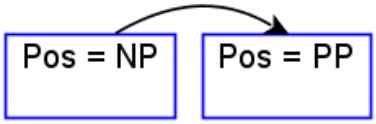

Figure 3: Graphical version of the example query (6) with a `NP` node dominating a `PP` node, taken from the ICARUS query editor.

4 Required Query Concepts

In earlier work several requirement lists for corpus query tools have been compiled. Lai and Bird (2010) defined functional features needed to query treebanks (and as an extension any tree-like structure), such as hierarchy, constituency, temporal organization, boolean operations, closures and non-navigational requirements. On the matter of usability Mueller (2010) provided conditions such as, among others, answer time, maintenance cost and the management of search results. Purely technical (external) factors such as scalability, computational complexity, platform-independence, extensibility and interoperability also play a major role (Lai and Bird, 2004; Kepser, 2003).

Besides the systems mentioned in Section 3 we looked at several other CQL/CQS instances: CWB (Evert and Hardie, 2011), VIQTORYA (Steiner and Kallmeyer, 2002), FSQ (Kepser, 2003), COSMAS II (Bodmer, 2005), MonaSearch (Maryns and Kepser, 2009), TigerSearch (Lezius, 2002), MATE (Heid and Mengel, 1999), Fangorn (Ghodke and Bird, 2012), Poliqarp (Janus and Przepiórkowski, 2007), LPath (Lai and Bird, 2005), TGrep2 (Rohde, 2001), Emu (Cassidy and Harrington, 2001), PML-TQ (Pajas and Štěpánek, 2009), KoralQuery (Bingel and Diewald, 2015), Emdros (Petersen, 2004), NetGraph (Mírovský, 2006), SETS (Luotolahti et al., 2015)

While evaluating those query tools, we also encountered some patterns that demanded the addition of certain new requirements to above lists. A selection of those newly added requirements is described in detail in the remainder of this section:

Readability. The majority of CQL instances make heavy use of special characters as operators or delimiters in their syntax. While this clearly allows for a more compact definition of a query in textual form, it also reduces the readability to a degree where it is almost impossible to understand a query at first sight (e.g Figure 1). This issue becomes even more pronounced when taking into account researchers as users who are quite capable of explaining the phenomena of interest, but are unfamiliar with those kinds of syntax styles.

Model Limitations. When evaluating the effective expressiveness of a particular CQL, there are essentially three interrelated domains involved:

target as the universe for which the query is to be defined or which the CQS is meant to address. Fundamentally this comprises the entire variability of linguistic structures and annotations, regardless of modality or theory. In addition this domain defines the upper bound of what can be directly queried by any system.

model as the intermediary part of the target domain which is covered by the CQL's underlying data model. This domain plays a crucial role as data models for CQSs tend to be geared towards a specific resource or a predefined set of phenomena. As such we often see query languages emerge that allow users to only extract very little of what the original target domain has to offer.

query as the resulting part of the model domain that can be addressed when defining a query. Limiting factors for this domain are often technical in nature (for example tractability concerns based on the desired query engine) or directly imposed by the model domain.

In set theory this leads to the expression $querydomain \subseteq modeldomain \subseteq targetdomain$. If we assume that modeling approaches like SALT or our recently proposed modeling framework (Gärtner and Kuhn, 2018) offer enough flexibility to maximize the modeled part of the target domain, then there should be no reason to limit a CQL up front to only a small subset of what can be modeled in a way that would later impact extensibility.

Flexible Scope. Query languages for database systems usually feature some sort of scope declaration that defines what part of the database a query is interested in, structurally and content wise (e.g. the obligatory `FROM` section in an SQL query). CQLs on the other hand lack this concept completely and often are bound to a fixed corpus structure, such as a treebank. But from a technical perspective there is no structural difference between things like a dependency or coreference tree:

As long as the user can specify which of the available structures he wishes to address, the QL should allow the same constraints to work on all structures or phenomena that share a common type. Our recently proposed approach to modeling corpora (Gärtner and Kuhn, 2018) introduced the concept of combining general-purpose data structures similar to graph models with linguistically motivated metadata that describes the composition, content and dependencies of a corpus. We believe this to be a promising step towards implementing CQLs of increased flexibility as it streamlines access to very diverse corpus resources.

Postprocessing Directives. The average CQL today does not contain any options to influence how results should be processed. Exceptions are the ability in several CQL implementations to limit the overall number of result instances to be returned. Besides that only PML-TQ (Pajas and Štěpánek, 2009) provides a truly rich functionality for post-processing of search results.

Already Mueller (2010) named the handling of search results as the "Achilles heel of corpus query tools". Even simple directives could already yield a great increase in usability, for example customizable sorting of results or returning the n-best results according to a user-defined criterion instead of the usual first n.

Learning Curve. An often overlooked aspect of a CQS is the initial training required to properly use it. Obstacles on the road to mastering a CQS (or its CQL alone) are for example the complexity of the query language, translation of ones question into a valid query expression or even purely user interface related issues. And even after having overcome those initial challenges, users can still be faced with recurring problems related to individual corpora that they want to query with a given CQS: "How is phenomenon X expressed in the resource?" "What tag is used to encode category Y?"

While there are parts of a CQS (such as learning a formal language in order to exploit the full capabilities of the system) that will likely remain tedious for the user forever, the concept of example-based search can ease the associated learning curve into a more friendly shape.

Different approaches to this concept, where users don't need a deeper understanding of technical details or the query language involved to get started, have been implemented in existing CQS:

GrETEL [10](Augustinus et al., 2012) allows to query a treebank by providing (parts of) an example sentence containing the syntactic structure one is interested in. It then internally parses the text and creates a query based on what the user declared to be relevant in the sentence snippet, to run against the treebank data. This effectively shields users from the query language overload and provides rapid results with minimal learning effort.

In **ICARUS** we integrated a parser pipeline (which is customizable via provision of fully trained models) that the user can employ to create "query templates" from the parser output based again on example input containing the phenomenon in question. Subsequent relaxation of the query (that is, deleting irrelevant parts of the query or

[10]Greedy Extraction of Trees for Empirical Linguistics

making them less strict) will then eventually yield a query that is general enough to find other instances of the same phenomenon. Since the treebank visualization and graphical query editor share many of their visual features, it is relatively easy to produce even complex queries from this process.

5 Syntax Design

In this section we introduce the current draft of our CQL design. To fulfill the added requirements from Section 4, we decided to partly move away from the classic CQL syntax style variants, one of which is shown in Figure 1. We base our design on two assumptions that seem reasonable to us, but which still need actual user studies for testing:

Prevalence of graphical query construction. With the ever increasing richness of annotated corpora, the user will require elaborate visualizations to comprehend their content. This becomes even more pronounced when taking into account multimodal corpora or multiple concurrent structural annotations, such as syntax trees. It seems likely that the user will then be more inclined to use graphical means of constructing queries, assuming their visual appearance is very similar to the way in which the respective phenomena were presented.

Feedback from researchers using our ICARUS tool indicated a strong preference towards the graphical part of the query editor once they became familiar with the system. However, this indication might be biased due to the graphical editor providing more contextual help (such as listing possible operators and in some cases attributes that can be extracted for querying) and requires further investigation.

Preference for Readability. This is in some way a follow-up to the first assumption above. If we assume the complexity of questions posed by researchers to increase further, then the resulting queries are bound to increase in size and complexity accordingly. Since even simple queries in many CQL implementations are already difficult to decipher, overly confusing query constructs that result from more complex questions will make the use of those systems prohibitive for a large portion of the target group. The essence of our assumption here is that a CQL that is more verbose, but also closer resembling the formulation of a question in natural language, will be preferred over the more artificial ones, even if the total length of the query increases. The added effort for constructing such a longer query can be mitigated in large parts when users truly focus on creating queries graphically. In that case boilerplate code in the query (comparable to the recurring SQL pattern "`FROM` x `SELECT` y `WHERE` z") can be handled by the application logic that drives the user interface.

As a result we envision a hybrid solution that incorporates aspects from both SQL and some successful existing CQL implementation. At its core is a division of queries into parts with different responsibilities, such as a scope preamble, structural sub-queries with local constraints (following the popular bracketing style), global constraints between those sub-queries and directives for pre- or postprocessing. Following the SQL concept, operators and delimiters between those parts appear in a more intuitive form as shown in a preliminary mock-up in Figure 2. We chose the bracketing style as opposed to logical terms to express structure and grouping of local constraints, as it is equally suitable for structural queries and those only targeting variable word sequences.

6 Conclusion

In this paper we discussed and conceptually evaluated existing CQL approaches, with a strong focus on basic usability. This was done in an ongoing project where CQS and CQL solutions are needed for accessing richly annotated and very diverse multi-modal corpora. As a first result we found current CQL instances to be unfit to meet the needs of our target group of technically less skilled users that might shy away from many existing complex CQL options. We then proceeded to compile additional requirements for such CQL implementations besides the previously established lists of requirements. Finally we proposed our draft of a hybrid approach to combine elements of increased usability with traditional CQL concepts. As a next step we plan to conduct user studies in order to verify some of the assumptions our design idea is based on and to refine the specification so we can implement a prototype system for further testing.

Acknowledgments

This work was funded by the German Federal Ministry of Education and Research (BMBF) via CLARIN-D, No. 01UG1120F and the German Research Foundation (DFG) via the SFB 732, project INF.

References

Liesbeth Augustinus, Vincent Vandeghinste, and Frank Van Eynde. 2012. Example-based treebank querying. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 3161–3167, Istanbul, Turkey, May. European Language Resources Association (ELRA). ACL Anthology Identifier: L12-1442.

Piotr Banski, Elena Frick, and Andreas Witt. 2016. Corpus query lingua franca (cqlf). In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Joachim Bingel and Nils Diewald, 2015. *KoralQuery – A General Corpus Query Protocol*, volume 111, pages 1–5. Linköping University Electronic Press.

Franck Bodmer. 2005. Cosmas ii - recherchieren in den korpora des ids. *Sprachreport : Informationen und Meinungen zur deutschen Sprache*, 21(3):2 – 5.

Aljoscha Burchardt, Sebastian Padó, Dennis Spohr, Anette Frank, and Ulrich Heid. 2008. Constructing integrated corpus and lexicon models for multi-layer annotations in OWL DL. *Linguistic Issues in Language Technology*, 1:1–33.

S. Cassidy and J. Harrington. 2001. Multilevel annotation in the EMU speech database management system. *Speech communication*, 33(1-2):61–78.

Stefan Evert and Andrew Hardie. 2011. Twenty-first century Corpus Workbench: Updating a query architecture for the new millennium. In *Proceedings of the Corpus Linguistics 2011 conference*, Birmingham.

Elena Frick, Carsten Schnober, and Piotr Bański. 2012. Evaluating query languages for a corpus processing system. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

Markus Gärtner, Gregor Thiele, Wolfgang Seeker, Anders Björkelund, and Jonas Kuhn. 2013. ICARUS – an extensible graphical search tool for dependency treebanks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Sofia, Bulgaria, August. Association for Computational Linguistics.

Markus Gärtner, Katrin Schweitzer, Kerstin Eckart, and Jonas Kuhn. 2015. Multi-modal visualization and search for text and prosody annotations. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 25–30, Beijing, China, July. Association for Computational Linguistics and The Asian Federation of Natural Language Processing.

Sumukh Ghodke and Steven Bird. 2012. Fangorn: A system for querying very large treebanks. In *COLING 2012: Demonstration Papers*, pages 175–182, Mumbai, India, December.

Markus Gärtner and Jonas Kuhn. 2018. A lightweight modeling middleware for corpus processing. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, may. European Language Resources Association (ELRA).

Ulrich Heid and Andreas Mengel. 1999. Query language for research in phonetics. In *International Congress of Phonetic Sciences (ICPhS 99)*, pages 1225–1228, San Francisco, August.

Daniel Janus and Adam Przepiórkowski. 2007. Poliqarp: An open source corpus indexer and search engine with syntactic extensions. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 85–88, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matthias Jarke and Yannis Vassiliou. 1985. A framework for choosing a database query language. *ACM Comput. Surv.*, 17(3):313–340, September.

Stephan Kepser. 2003. Finite structure query: A tool for querying syntactically annotated corpora. In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, pages 179–186, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thomas Krause and Amir Zeldes. 2014. Annis3: A new architecture for generic corpus query and visualization. *Digital Scholarship in the Humanities*.

Thomas Krause, Ulf Leser, and Anke Lüdeling. 2016. graphANNIS: A Fast Query Engine for Deeply Annotated Linguistic Corpora. *JLCL*, 31(1):iii–25.

Catherine Lai and Steven Bird. 2004. Querying and updating treebanks: A critical survey and requirements analysis. In *In Proceedings of the Australasian Language Technology Workshop*, pages 139–146.

Catherine Lai and Steven Bird, 2005. *LPath+: A First-Order Complete Language for Linguistic Tree Query*. ACL Anthology, 12.

Catherine Lai and Steven Bird. 2010. Querying linguistic trees. *J. of Logic, Lang. and Inf.*, 19(1):53–73, January.

Wolfgang Lezius. 2002. *Ein Suchwerkzeug für syntaktisch annotierte Textkorpora*. Ph.D. thesis, IMS, University of Stuttgart. Arbeitspapiere des Instituts für Maschinelle Sprachverarbeitung (AIMS), volume 8, number 4.

Juhani Luotolahti, Jenna Kanerva, Sampo Pyysalo, and Filip Ginter. 2015. Sets: Scalable and efficient tree search in dependency graphs. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 51–55, Denver, Colorado, June. Association for Computational Linguistics.

Hendrik Maryns and Stephan Kepser. 2009. Monasearch: Querying linguistic treebanks with monadic second-order logic. In *The 7th International Workshop on Treebanks and Linguistic Theories*.

Jiří Mírovský. 2006. Netgraph: A tool for searching in prague dependency treebank 2.0. In Jan Hajič and Joakim Nivre, editors, *Proceedings of TLT 2006*, pages 211–222, Praha, Czechia. ÚFAL MFF UK.

Martin Mueller. 2010. Towards a digital carrel: A report about corpus query tools.

Petr Pajas and Jan Štěpánek. 2009. System for Querying Syntactically Annotated Corpora. In *ACL-IJCNLP: Software Demonstrations*, pages 33–36, Suntec, Singapore.

Ulrik Petersen. 2004. Emdros: A text database engine for analyzed or annotated text. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Douglas L.T. Rohde. 2001. TGrep2 the next-generation search engine for parse trees. http://tedlab.mit.edu/ dr/Tgrep2/.

Ilona Steiner and Laura Kallmeyer. 2002. Viqtorya - a visual query tool for syntactically annotated corpora.

Florian Zipser. 2009. Entwicklung eines Konverterframeworks für linguistisch annotierte Daten auf Basis eines gemeinsamen (Meta-)modells, November. In this diploma thesis I present a framework to convert linguistic data coming from a specific linguistic format to other linguistic formats. This approach is based on a common meta-model, which is used as an intermediate step to decrease the number of necessary mappings.